TEXAS A&M UNIVERSITY

Hardware Implementation of a Desktop Supercomputer for High Performance Image Processing

ONR Grant Number N00014–94–1–0516

Technical Report
May/01/95 – Aug/01/95

DEPARTMENT OF ELECTRICAL ENGINEERING
College Station, Texas

Hardware Implementation of a Desktop Supercomputer for High Performance Image Processing

ONR Grant Number N00014–94–1–0516

Technical Report
May/01/95 – Aug/01/95

Feasibility study of a Time–Multiplexing Approach using a 4x4 CNN Laboratory Prototype

Dr. Jose Pineda de Gyvez

Texas A&M University
Microelectronics Group
Department of Electrical Engineering
College Station, TX, 77843

Phone: (409) 8457477
FAX: (409) 8457161
Email:gyvez@pineda.tamu.edu

I. INTRODUCTION

This report adresses the implementation of a time-multiplexing scheme using a 4x4 discrete implementation of a CNN. The objective is to demonstrate the feasibility of using a small CNN array to process a large image using an actual analog CNN.

In the research and development of neural network circuits, tools are necessary for effective simulation, instrumentation, and measurement. In particular, image-processing applications using cellular neural networks [1] need dedicated hardware and software for graphics-intensive I/O. The user should be able to easily enter and modify data for experimentation. Control commands are also needed to initialize, process, and collect results from an expandable array of cells. Finally, the final results should be presented to the researcher in a meaningful form.

A software/hardware combination is proposed here as a solution to this problem, as shown in Figure 1.

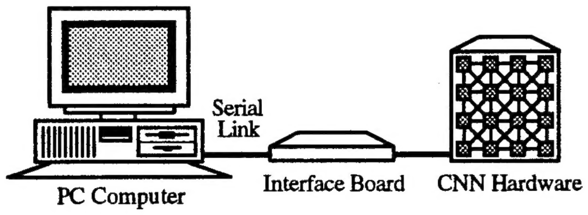


Figure 1. Complete hardware setup

It consists of a PC linked to a microcontroller interface board, which can be attached to different neural network configurations. The controlling software uses a Windows™ graphical user interface (GUI), enabling the creation of custom software that is both intuitive and easy-to-use. By adding a high level of abstraction between software and hardware, the user is shielded from the complexity of the neural network implementation's details. In this experimental prototype the transfer of data by the PC-CNN interface is carried out via an RS-232 serial link. The interfacing hardware is capable of converting data input by the user into analog voltages, which can be distributed to any node of the CNN. Results from the network are converted to binary numbers and relayed back to the PC software for display as a bitmapped image. Though a CNN image-processing hardware is presented in this report, other network topologies will work as well. The CNN simulator for PC's follows the theory reported elsewhere and some convergenge criterion of [2] and [3].

II. CNN SOFTWARE AND GRAPHICAL USER INTERFACE

The Microsoft Windows™ GUI provides the user an easy way to control the CNN hardware, without having to deal with low-level system routines. Both hardware and software CNN processing options are provided to allow the comparison of results. This is particularly useful for benchmarking the accuracy of neural network hardware during its development process. Figure 2 illustrates the various menu options.

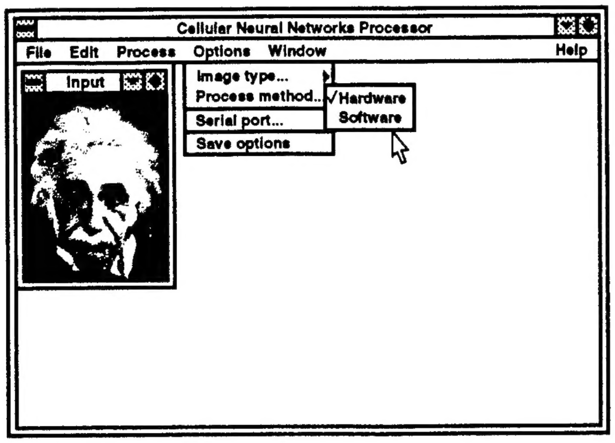


Figure 2. Software interface showing a loaded bitmap image

The File menu allows the user to load standard .BMP bitmap images for processing. After images are processed, they can be saved to the hard-drive or to a floppy disk. Similarly, the CNN templates can be loaded and saved, using files with the .TEM extension.

The Edit menu of the PC-CNN simulator contains selections for the pre-processing of images and templates. The template editor, shown in Figure 3, allows the modification of the **A** and **B** templates, the input bias, and the initial conditions. An auto-save feature can be set to automatically save templates after being edited. An Add Noise option randomly scatters

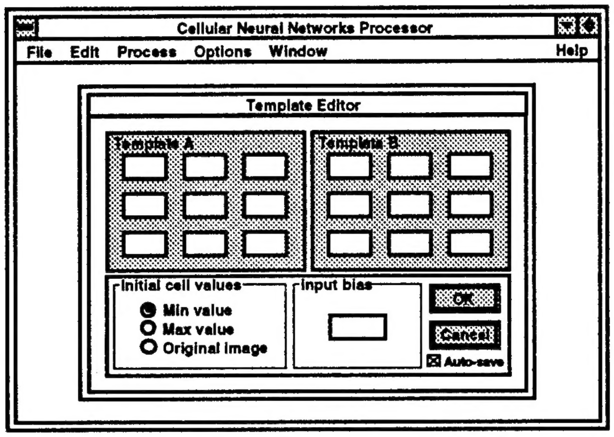


Figure 3. The template editor of the PC-CNN simulator

black and white pixels throughout the input image to allow testing of noise-removal templates. This option can be repeatedly selected as desired without permanently changing the original bitmap file.

Clicking on the Process menu initiates the CNN array processing. An hourglass cursor indicates that the user must wait for a duration of time, which depends on the size of the loaded image. The program can also detect the absence of the neural network hardware, and will alert the user immediately after an attempt at hardware processing.

An Options menu enables the user to configure the processing and I/O settings. The Image Type option contains choices for black-and-white or color processing (the latter is for future development). The Process Method option sets the program's mode for either hardware processing or software emulation. As mentioned earlier, this feature is good for comparing hardware processing with known software results.

Serial port settings (COM port, baud rate, parity, etc.) can be configured to work with most PC-compatible computers.

After an input image and template is loaded, processing the image is as simple as selecting the Process menu option. All necessary information is sent to the CNN, and is returned when completed. Finally, an output window quickly appears to reveal the newly processed image.

III. SERIAL COMMUNICATIONS PROTOCOL

The transfer of data between the PC and interfacing hardware uses an error-detection protocol that has three levels of protection. This was developed to prevent possible noise during transmission, which can alter the processed image. This can be a problem, especially when using noise-removal templates.

The first method of detection uses bit parity-checking that can detect simple errors in the data bits. The parity can be either even or odd, as defined in the RS-232 standard. Since this will only detect bytes that were changed, additional methods were employed to detect missing bytes. The second method uses frequent handshaking signal exchanges to confirm that all data has been received. The PC and interface board are always kept in constant communication, enabling each to know what the other is doing. The third method is a "time-out" feature that aborts the transfer after a certain period of inactivity. This prevents the possibility of errors caused by corrupted handshaking signals, as well as disconnected hardware.

In Figure 4, the protocol flowchart of the PC-to-CNN transfer is shown. The interface board initially waits in a loop, polling for a *SEND* request from the PC. The board responds with an *OK* signal to confirm that the command has been received. The templates,
images, and initial conditions are sent in packets to keep all data synchronized. This accomplished by beginning and ending each packet with *START* and *END* signals. This prevents missing data from being incorrectly shifted into subsequent data packets. Any time one of these handshakes are not fully completed, the protocol aborts the transfer and waits for the *SEND* command again. This will also detect any corrupted signals. For the CNN-to-PC direction of transfer, the protocol is reversed in a similar manner.

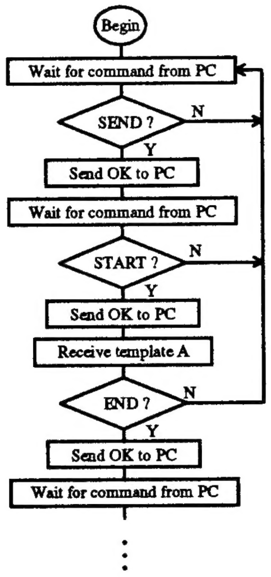


Figure 4. PC-to-CNN communications protocol flowchart

IV. CNN CELL HARDWARE

A time-multiplexing CNN scheme [2], shown in Figure 5, was used to limit the amount required hardware. It consists of a 4x4 array of identical cells that collectively perform analog neural processing.

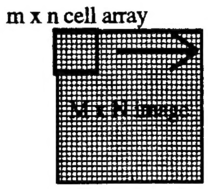


Figure 5. Time-multiplexing a large image

Each cell is attached to its own small motherboard which is joined to other motherboards via ribbon cable. Figure 6 illustrates how the symmetry of the motherboard's open-architecture design cleverly routes neighboring signals to their correct destination. For the case of signals between diagonal cells, each motherboard uses adjacent boards as stepping-stones to carry the signal diagonally. This makes the task of expanding the network as simple as "plugging-in" more cells. Each CNN cell consists of eighteen analog multipliers, an opamp, and an analog multiplexer, shown in Figure 7. Fig. 8 displays an actual photograph of one cell card.

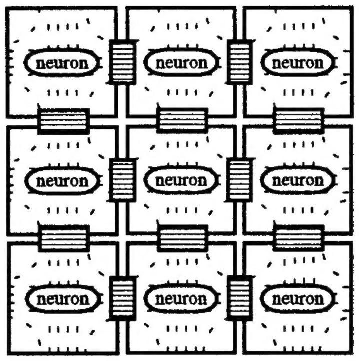


Figure 6. Modular cell design

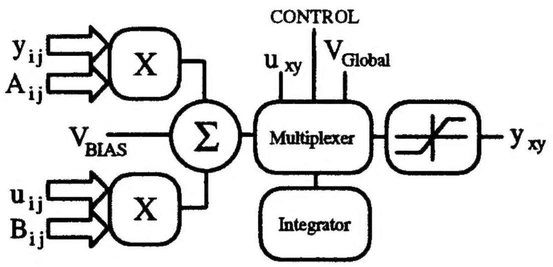


Figure 7. CNN cell block diagram

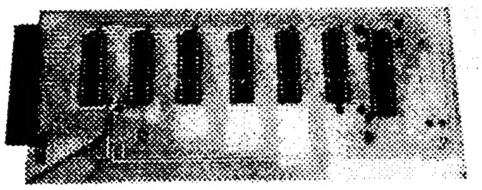


Fig. 8. (a)CNN cell card. (b) 4x4 Full discrete implementation

The multipliers are used to multiply the weights of the **A** and **B** templates with the feedback **y** and input **u**, respectively. These results are sent through a lossy RC integrator that controls the neuron's convergence time. The opamp is used as the thresholding activation function

$$y_{ij} = f(x_{ij})$$

where x_{ij} is the current state of cell (i, j). There are four modes of operation that is controlled by the switching of the multiplexer. The first initializes the state of all cells to their corresponding input pixel value, at the time-multiplexing window's present position within the image. The second mode initializes all cells to a user-defined global state, which can be either black or white. The third mode switches-in the integrators for all of the cells to begin the neural processing. The final mode disconnects the interactions among cells to hold their final converged values for retrieval.

A global voltage bias is sent to all of the cells to provide an adjustable offset that is part of the user-defined templates (i.e. edge detection, hole filler, etc.). This is used instead of a current bias because the summing of all neighboring interactions are performed in the voltage domain. Once the processing is complete, the analog voltage outputs of each cell are converted to a binary numbers by the CNN interfacing hardware.

Although the hardware can currently process black-and-white, a provision has been designed to easily upgrade the circuit to handle color images. The output **y** would simply be replaced by the state **x**, to allow intermediate voltages to simulate varying pixel intensities.

V. INTERFACING HARDWARE

In order for the CNN hardware to receive pixel values, special circuitry is needed to latch and hold data sent by the PC. It must also provide a way to convert and return the image to be displayed.

The interfacing hardware consists of a 16-bit Motorola HC16 microcontroller attached to decoding circuitry, as in Figure 9. The HC16 data bus is decoded into 66 unique select lines, 64 of which are used to initialize individual sample-and-hold chips. The remaining two control lines are used to select the CNN's mode of operation, as described earlier. The sample-and-hold chips contain voltage values that

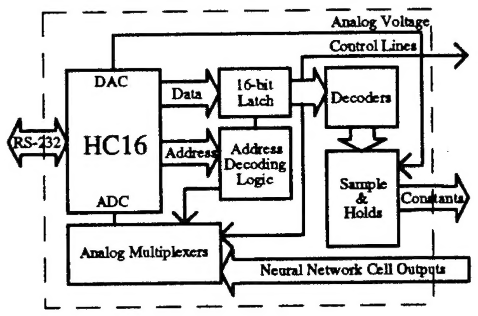


Figure 8. Interfacing hardware diagram

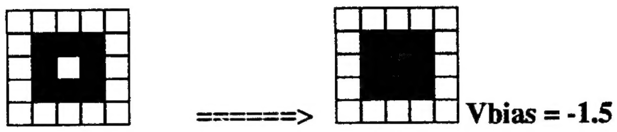
must remain constant for the duration of each time-multiplex: 1) template **A**, 2) template **B**, 3) V_{bias}, 4) input image **u**, and 5) the voltage V_{Global}. A static border around the multiplex window is also assigned pixels from the input image to assist in the overlapping of multiplexes, minimizing the error caused by missing neighbor cells. Each line must be selected one-at-a-time to share the HC16's single 16-bit D/A converter. Analog representations of the required constant voltage values are output and stored in the sample-and-hold chips for the CNN hardware to access.

There are sixteen 4-to-1 multiplexers, which can select cell output voltages in groups of eight back to the HC16's 8-channel, 8-bit A/D converter.

The microcontroller software is stored in two 32k EPROMs, and is used to manage the time-multiplexed processing and to communicate with the PC. This serial link is established using the standard RS-232 cable, communicating at a speed of 19,200 bits-per-second (bps). After the PC sends the images, templates, initial conditions, the multiplexing routines sends the necessary data to the CNN through the D/A converter and waits until the cells converge. The A/D converter returns the converged CNN pixel values. The processed pixels are then stored in a separate address of memory, as the multiplex window scans across the entire original image. When the processing is complete, the microcontroller sends the resulting image to the PC using the serial transfer protocol.

VI. BENCHMARKING

A simple 3 x 3 box was used to measure the convergence time of a *Hole-Filler* template
The center cell's voltage transient was probed with a digitizing oscilloscope. In a hole-filler, any white pixel surrounded by a majoriy of black pixels will become black. In an *Edge-Detection* template, the opposite is true; it removes black pixels from solid regions, leaving an outline of the picture. The plots of Figures 10b and 11b show the convergence times for a cell under the described conditions.



Template A

0	1	0
1	4	1
0	1	0

Template B

0	0	0
0	4	0
0	0	0

Figure 10a. Hole-filler template

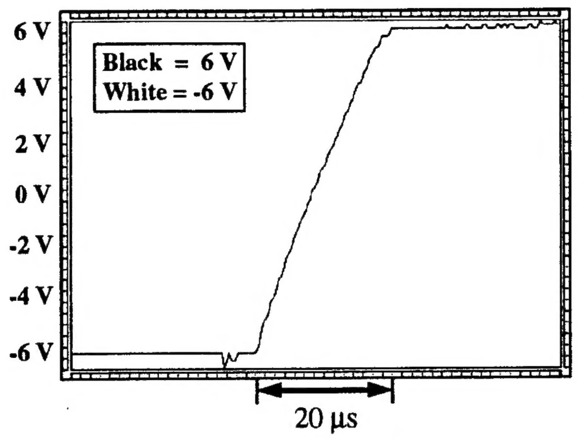
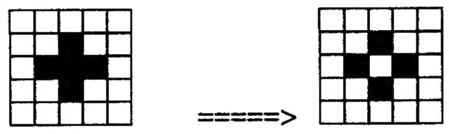


Figure 10b. Hole-filler output voltage transient



Template A

0	0	0
0	4	0
0	0	0

Template B

-1	-1	-1
-1	4	-1
-1	-1	-1

Vbias = -2.5

Figure 11a. Edge-detection template

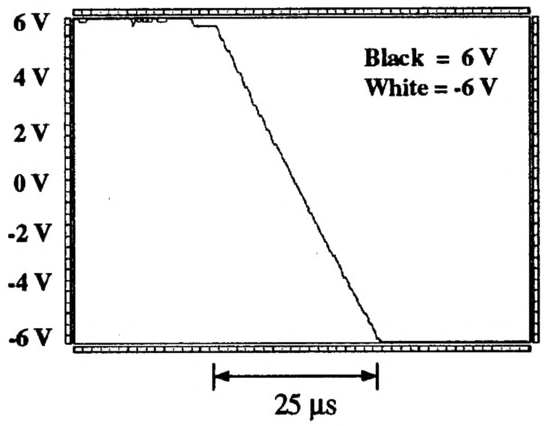


Figure 11b. Edge-detection output voltage transient

VII. IMAGE PROCESSING RESULTS

Figures 12 and 13 demonstrate the image processing capabilites of cellular neural networks to highlight key features of military satellite photos. A unprocessed noisy image containing roads and buildings is shown in Figure 12a, and an aerial photo of ships at a dock is shown in Figure 13a. The images were processed to highlight human-made objects from natural objects. Using a noise-removal template in software simulations gives their corresponding results in Figures 12b and 13b, respectively. Figures 12c-e and 13c-e shows the application of the CNN hardware with variations of the time-multiplexing scheme. Tables 1 and 2 provide a brief summary of processing times and specs for the prototype, respectively.

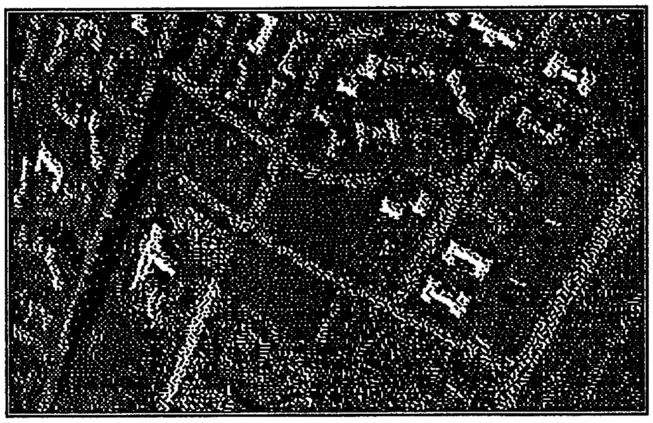


Figure 12a. Target #1

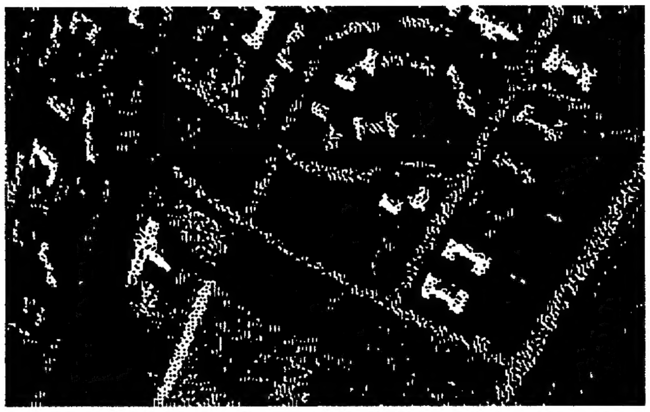


Figure 12b. Target #1 (software simulation)

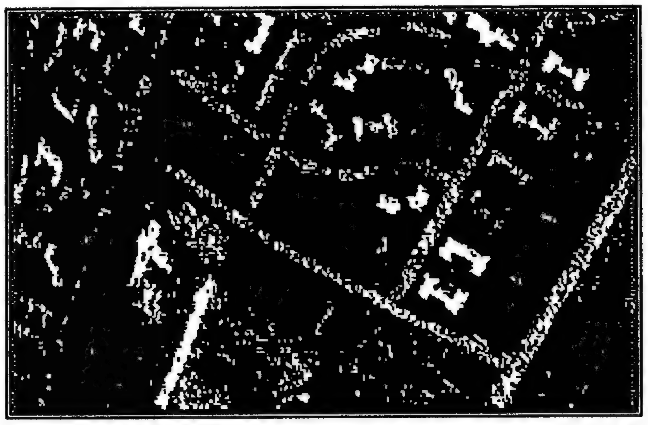


Figure 12c. Target #1 (3x3 CNN with no overlap)

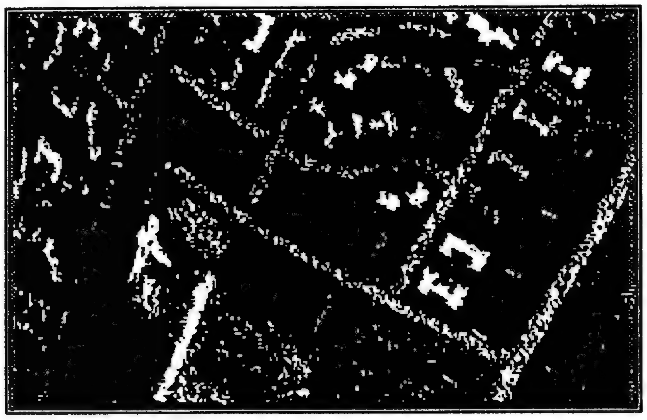


Figure 12d. Target #1 (4x4 CNN w/ overlap of 1 cell)

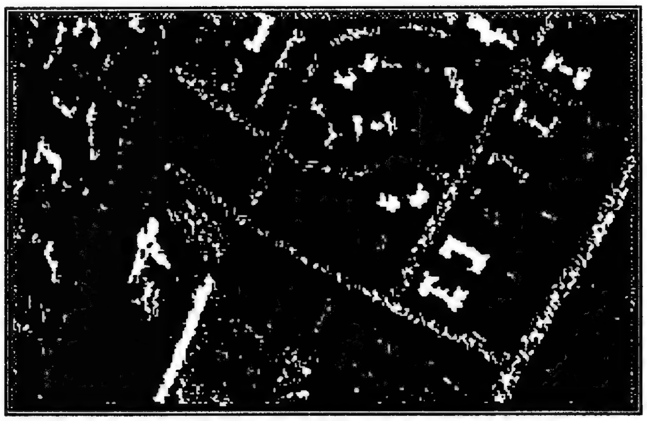


Figure 12e. Target #1 (3x3 CNN w/ overlap of 1 cell)

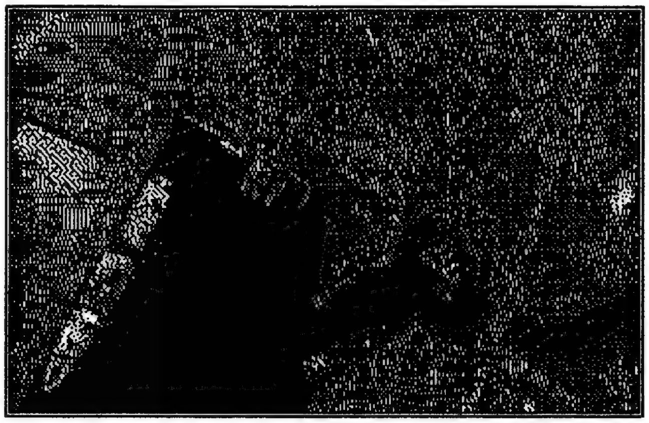


Figure 13a. Target #2

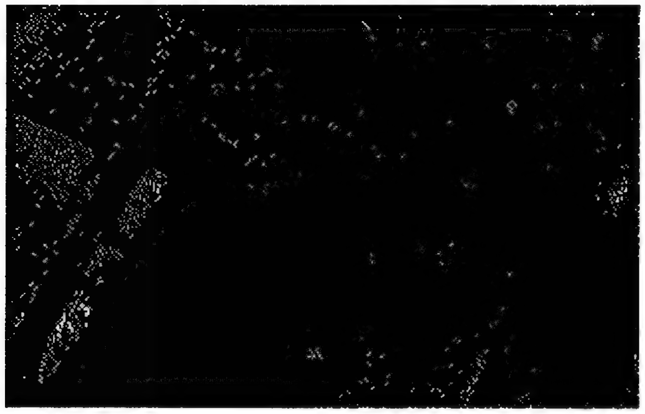


Figure 13b. Target #2 (software simulation)

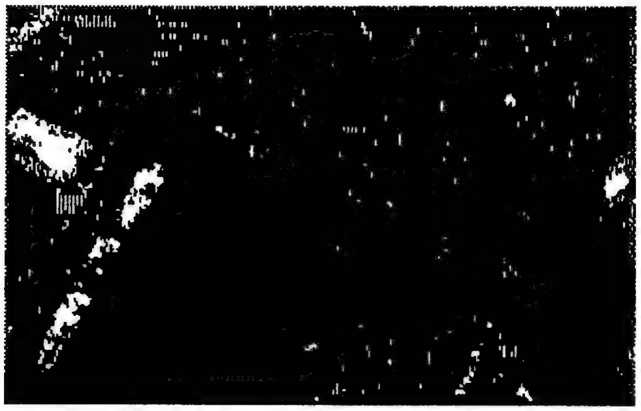


Figure 13c. Target #2 (3x3 CNN with no overlap)

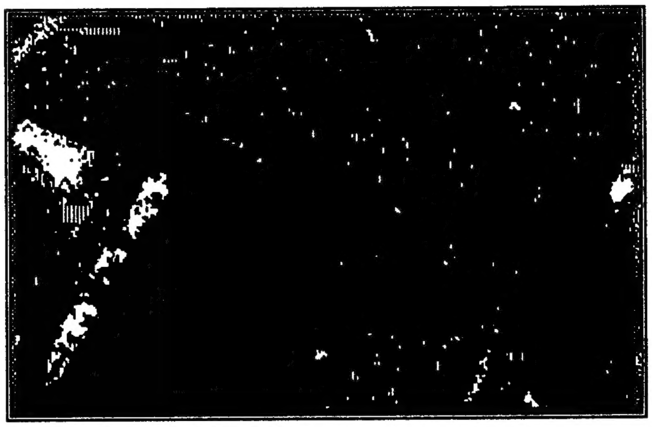


Figure 13d. Target #2 (4x4 CNN w/ overlap of 1 cell)

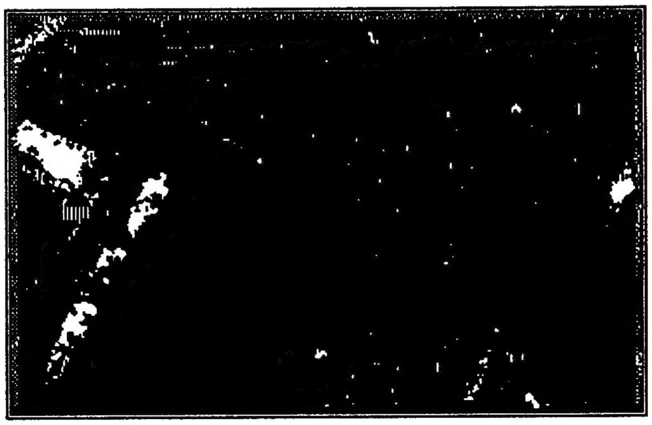


Figure 13e. Target #2 (3x3 CNN w/ overlap of 1 cell)

Table 1. CNN Processing times

Figure	Execution Time	Comments
12b	7.762 minutes	software simulation
12c	3.070 minutes	hardware is 153% faster
12d	4.648 minutes	hardware is 67% faster
12e	13.017 minutes	slower, but with less error

Table 2. CNN spec summary

Max image size (HW mode)	No limit*
Max image size (SW mode)	No limit*
Power dissipation of CNN	≈ 1 watt/cell
Voltage supplies	+5, ±7, ±15V
Convergence parameter τ	0.5 µs
Slope of f(x)	2
Max serial transfer rate	19,200 bps

*Limited by tbe amount of available RAM on the PC.

VII. CONCLUSION

An integrated system for image procfessing is proposed. The main elements of the system consist of a CNN software simulator, multiplexing hardware, and an interface between a PC and the hardware and software. From the results, it can be seen that the CNN hardware is much faster than CNN software simulators. Even with no multiplexing window overlap, the processing speed is increased without a noticable loss of image quality when compared with the "overlapped" results. The processed output demonstrates the potential of the time multiplexing scheme for processing large images. The current slow processing is due to the serial interface and to the 16mhz HC1616 microcontroller. This shortcoming will be fixed by using a high speed I/O board with Direct Access Memory.

VIII. REFERENCES

[1] L.O. Chua and L. Yang, "Cellular Neural Networks: Theory, "IEEE Trans. Circuits Systems, vol. 35, pp.1257-1272, 1988.

[2] C.C. Lee and J. Pineda de Gyvez, "Time-Multiplexing CNN Simulator," Proceedings of Intl. Symposium on Circuits and Systems, pp. 407-410, May 1994.

[3] J.A. Nossek and T. Roska, editors. "Special Issue on Cellular Neural Networks," *IEEE Trans. Circuits and Systems*, vol. 40, March 1993.

[4] L.O. Chua and L. Yang, "Cellular Neural Networks: Applications," *IEEE Trans. Circuits and Systems*, vol 35, pp. 1273-1290, 1988.

[5] K. Slot, Determination of cellular neural network parameters for the feature detection of two dimensional images," *Proc. IEEE Int. Workshop on Cellular Neural Networks and Their Applications*, pp. 82-91, 1990.

[6] L.O. Chua and P. Thiran, "An analytic method for designing simple cellular neural networks," *IEEE Trans. Circuits and Systems*, vol 38, pp. 1332-1341, 1991.

[7] L.O. Chua and T. Roska, "The CNN Paradigm," *IEEE Trans. on Circuits and Systems -1: Fundamental Theory and Applications*, vol 40, no. 3, pp. 147-156, March 1993.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** August 14, 1995	**3. REPORT TYPE AND DATES COVERED** Technical Report 5/01/95 - 8/01/95

4. TITLE AND SUBTITLE
Feasibility study of a Time Multiplexing Approach using a 4x4 CNN Laboratory Prototype

5. FUNDING NUMBERS
G
N99914-94-1-0516

6. AUTHOR(S)
Jose Pineda de Gyvez

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Texas Engineering Experiment Station
Texas A&M University

8. PERFORMING ORGANIZATION REPORT NUMBER
93-549/#6

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
Code 214: JWK
Ballston Tower One
800 North Quincy Street
Arlington, Virginia 22217-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
A: Approved for public release: distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The state of the art work has concentrated on VLSI CNN implementations without really addressing the "system level". While efficient implementations have been reported, no reports have been presented on the use of these implementations for processing large complex images. This report presents a feasibility study to use our time multiplexing scheme based on a CNN prototype. The laboratory prototype consists of a 4x4 CNN analog discrete implementation serially linked to a PC. A simple graphical interface has been built to be able to read bitmap images of any size and to send them to the CNN. The results are encouraging as they demonstrate the potential of the multiplexing approach applied to complex black and white images. These images consist of two aerial views of 256x256 pixels processed only with the 4x4 CNN.

14. SUBJECT TERMS
Cellular Neural Networks, VLSI, Multiplexing

15. NUMBER OF PAGES
12

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT**

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to ***stay within the lines*** to meet ***optical scanning requirements***.

Block 1. Agency Use Only *(Leave blank)*.

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	Contract	PR	Project
G	Grant	TA	Task
PE	Program Element	WU	Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of...; To be published in.... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	See authorities.
NASA	See Handbook NHB 2200.2.
NTIS	Leave blank.

Block 12b. Distribution Code.

DOD	Leave blank.
DOE	Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	Leave blank.
NTIS	Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17. - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.